Backend-Driven UI: Making Screens Dynamic

Self-intro

- Name: Joshua Kaplan (@yhkaplan)
- Work: minne @ GMO Pepabo
- Interests: 🤖CI/CD, 📦 frameworks, and more
- Hobbies: 🥯bread, 📚history, and 🏃running

What is a backend driven UI?

— Extreme end: every individual view defined by backend

— Less extreme: order of and type of view defined by backend

— Why JSON?

Purpose

- Change content for each customer
- A/B testing
- Feature flags
- Less work to make changes

Demo

How I made it

- Prototype in sample app
- Use compositional layout and diffable data sources
- Define each section type as JSON w/ a title and subtitle
- Firebase Remote Config

Code Example

```
let homeReducer = Reducer<HomeState, HomeAction, HomeEnvironment> { state, action, environment in
    switch action {
    case .loadSectionData:
        state.isSectionLoading = true
        return environment.homeService.homeContentPublisher()
            .replaceError(with: [])
            .receive(on: DispatchQueue.main)
            .map { HomeAction.loadItemData(sections: $0) }
            .eraseToEffect()

    case let .loadItemData(sections):
        state.isSectionLoading = true
        return environment.homeService.sectionItemsPublisher(sections: sections)
            .replaceError(with: [:])
            .receive(on: DispatchQueue.main)
            .map { HomeAction.setSections(sections: $0) }
            .eraseToEffect()
    }
    return .none
}
```

Challenges/risks

- App Review risks (AKA don’t pull a Fortnite)
- Server unavailable

Other possible approaches

- Defining more in JSON
- microsoft/AdaptiveCards
- spotify/HubFramework
- Web views

Conclusion

— Easy to strike a balance w/ compositional layout
— Think of important, frequently changing screens
— Worth it for minne
— Try it out in a prototype!
— yhkaplan/Shop